PATENT
IBM Docket No. GB920000081US1

In the United States Patent and Trademark Office

Date: August 21, 2001

In re Application of: Osborne, Andrew J., et al ***Filed:*** Herewith

For: Recognition of Command Related Items in Object Code

Serial Number: To be assigned.

Art Unit: Not yet assigned. ***Examiner:*** To be assigned

Preliminary Amendment

Hon. Commissioner of Patents and Trademarks
Washington, DC 20231

Sir:

Prior to the examination of the subject application, please amend the above mentioned application as follows:

In the Claims

CLAIMS

1. A method of recognizing command related items in a body of object code, said command related items corresponding to command names and/or associated option names from a textual programming language, said method comprising the steps of:

entering a list of entries, each comprising a required command name and/or option names in programming language textual form, into a filter table;

scanning the body of object code for all bit strings potentially representing command names and identifying such command names;

for each of the potential command names so identified, examining a number of succeeding bits for bits which represent valid options for each said potential command name to further identify commands having valid combinations of command names and options; and

for said identified commands, comparing said identified command names and/or option names in programming language textual form with the entries of said filter table to determine whether or not they match any of the list of required command names and/or options in said filter table.

2. A method as claimed in Claim 1, further comprising the step of, after said scanning and examining steps, validating syntax of each command comprising a command name followed by one or more valid option names and comparing only validated command names and/or option names with the entries in the filter table.

3. A method as claimed in Claim 2, wherein said validating step further comprises applying each said command to a syntax tree.

4. A method as claimed in Claim 1, wherein said filter table entries can specify both a presence and an absence of respective command names and/or option names in the scanned and examined object code.

5. A method as claimed in Claim 1, wherein at least some of said filter table entries include combinations of command and/or option names, said method further comprising the step of checking syntax of said combination entries to the filter table.

6. A method as claimed in Claim 1, wherein said scanning and examining steps involve comparing object code bit strings with bit strings extracted from a library which represent all possible command names and options for said programming language.

7. An object code recognition system for recognizing command related items in a body of object code, said command related items corresponding to command names and/or associated option names from a textual programming language, said system comprising:

a filter table for holding a list of entries, each comprising a required command name and/or option names in programming language form;

an object code scanner for scanning the body of object code for all bit strings potentially representing command names and identifying such command names, said scanner being arranged, in response to identification of each potential command name, to examine a number of succeeding bits for bits which represent valid options for each said command name to further identify commands having valid combinations of command names and options; and

a filter for comparing said identified command names and/or option names in programming language textual form with the entries of said filter table to determine whether or not they match any of the list of required command names and/or options in said filter table.

8. A system as claimed in Claim 7, further comprising a syntax checker for validating the syntax of each command, comprising a command name followed by one or more valid option names whereby only validated command names and/or option names are compared with the entries in the filter table by said filter.

9. A system as claimed in Claim 8, wherein said syntax checker includes a syntax tree.

10. A system as claimed in Claim 7, wherein said filter table entries can each specify both a presence and an absence of respective command names and/or option names in the scanned and examined object code, and said filter is responsive to said specification in determining whether or not said identified command names and/or option names match said filter table entries

11. A system as claimed in Claim 7, wherein at least some of said filter table entries include combinations of command and/or option names, said system further comprising means for checking the syntax of said combination entries to the filter table.

12. A system as claimed in Claim 7, further comprising a library containing bit strings representing all possible command names and options for said programming language.

13. A system as claimed in Claim 7, wherein said filter is arranged to generate a list of matching commands.

14. A computer program recorded on a medium and executable on a computer to recognise command related items in a body of object code, said command related items corresponding to command names and/or associated option names from a textual programming language, said program comprising:

a filter table data structure for holding a list of entries each comprising a required command name and/or option names in programming language form;

object code scanner code for scanning the body of object code for all bit strings potentially representing command names and identifying such command names, said scanner

code being arranged, in response to identification of each potential command name, to examine a number of succeeding bits for bits which represent valid options for each said command name to further identify commands having valid combinations of command names and options; and

filter code for comparing said identified command names and/or option names in programming language textual form with the entries of said filter table to determine whether or not they match any of the list of required command names and/or options in said filter table.

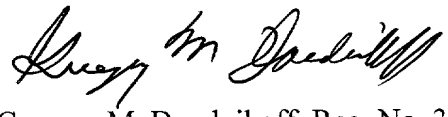
15. A computer program as claimed in Claim 14, wherein said object code scanner code includes verb objects for representing and identifying command names;

a parameter decoder object for decoding succeeding bits as potentially valid options on identified commands; and

a syntax object for validating the syntax of each command comprising an identified command name followed by one or more valid option names.

16. A computer program as claimed in Claim 15, further comprising a two dimensional array data structure, rows and columns of which are indexed by each of a pair of supplied bytes in said object code respectively, and a file parser object for supplying successive pairs of object code bytes to said array, the array comprising pointers to respective verb objects for each pair of supplied bytes representing a potentially valid command, the file parser object initiating respective verb objects in response to the return of a pointer from said array.

Respectfully Submitted,



Gregory M. Doudnikoff, Reg. No. 32,847
Attorney of Record

IBM Corporation
T81/503
PO Box 12195
Research Triangle Park, NC 27709
919-919-254-1288
FAX 919-254-4330

CLAIMS

1. A method of recognising command related items in a body of object code, said command related items corresponding to command names and/or associated option names from a textual programming language, said method comprising: the steps of

entering a list of entries, each comprising a required command name and/or option names in programming language textual form, into a filter table;

scanning the body of object code for all bit strings potentially representing command names and identifying such command names;

for each potential command name of the so identified, examining a number of succeeding bits for bits which represent valid options for each said command name to further identify commands having valid combinations of command names and options; and

for said identified commands, comparing said identified command names and/or option names in programming language textual form with the entries of said filter table to determine whether or not they match

any of the list of required command names and/or options in said filter table.

2. A method as claimed in claim 1, ~~including the~~ further comprising the step of after said scanning and examining steps, ~~of~~ validating ~~the~~ syntax of each command comprising a command name followed by one or more valid option names and comparing only validated command names and/or option names with the entries in the filter table.

3. A method as claimed in claim 2, ~~in which~~ wherein said validating step ~~of syntax validation~~ comprises further applying each said command to a syntax tree.

4. A method as claimed in claim 1, ~~in which~~ wherein said filter table entries can specify both ~~the~~ a presence and ~~the~~ an absence of respective command names and/or option names in the scanned and examined object code.

5. A method as claimed in claim 1, ~~in which~~ wherein at least some of said filter table entries include combinations of command and/or option names, said method further comprising the ~~further~~ step of checking ~~the~~ syntax of said combination entries to the filter table.

6. A method as claimed in claim 1, ~~in which~~ wherein said ~~the~~ scanning and examining steps involve comparing object code bit strings with bit strings extracted from a library which

represent all possible command names and options for said programming language.

7. An object code recognition system for recognising command related items in a body of object code, said command related items corresponding to command names and/or associated option names from a textual programming language, said system comprising:

a filter table for holding a list of entries, each comprising a required command name and/or option names in programming language form;

an object code scanner for scanning the body of object code for all bit strings potentially representing command names and identifying such command names, said scanner being arranged, in response to identification of each potential command name, to examine a number of succeeding bits for bits which represent valid options for each said command name to further identify commands having valid combinations of command names and options; and

a filter for comparing said identified command names and/or option names in programming language textual form with the entries of said filter table to determine

whether or not they match any of the list of required command names and/or options in said filter table.

8. A system as claimed in claim 7, ~~includes~~ further comprising a syntax checker for validating the syntax of each command, comprising a command name followed by one or more valid option names whereby only validated command names and/or option names are compared with the entries in the filter table by said filter.

9. A system as claimed in claim 8, wherein said syntax checker includes a syntax tree.

10. A system as claimed in claim 7, ~~in which~~ wherein said filter table entries can each specify both ~~the~~ a presence and ~~the~~ an absence of respective command names and/or option names in the scanned and examined object code, and said filter is responsive to said specification in determining whether or not said identified command names and/or option names match said filter table entries

11. A system as claimed in claim 7, ~~in which~~ wherein at least some of said filter table entries include combinations of command and/or option names, said system further comprising a means for checking the syntax of said combination entries to the filter table.

12. A system as claimed in claim 7, further comprising a library containing bit strings representing all possible command names and options for said programming language.

13. A system as claimed in claim 7, ~~in which~~ wherein said filter is arranged to generate a list of matching commands.

14. A computer program recorded on a medium and executable on a computer to recognise command related items in a body of object code, said command related items corresponding to command names and/or associated option names from a textual programming language, said ~~the~~ program comprising :

a filter table data structure for holding a list of entries each comprising a required command name and/or option names in programming language form;

object code scanner code for scanning the body of object code for all bit strings potentially representing command names and identifying such command names, said scanner code being arranged, in response to identification of each potential command name, to examine a number of succeeding bits for bits which represent valid options for each said command name to further identify commands having valid combinations of command names and options; and

filter code for comparing said identified command names and/or option names in programming language textual form with the entries of said filter table to determine whether or not they match any of the list of required command names and/or options in said filter table.

15. A computer program as claimed in claim 14, ~~in which~~ wherein said object code scanner code includes verb objects for representing and identifying command names;

a parameter decoder object for decoding succeeding bits as potentially valid options on identified commands; and

a syntax object for validating the syntax of each command comprising an identified command name followed by one or more valid option names.

16. A computer program as claimed in claim 15, further ~~including~~ comprising a two dimensional array data structure, ~~the~~ rows and columns of which are indexed by each of a pair of supplied bytes in said object code respectively, and a file parser object for supplying successive pairs of object code bytes to said array, the array comprising pointers to respective verb objects for each pair of supplied bytes representing a potentially valid command,

the file parser object initiating respective verb objects in response to the return of a pointer from said array.